Bounded single-machine parallel-batch scheduling with release dates and rejection

Lingfa Lu[1], **T.C.E. Cheng**[2,*], **Jinjiang Yuan**[1], **Liqi Zhang**[1]

1. Department of Mathematics, Zhengzhou University, Zhengzhou, Henan 450052, PR China
2. Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

Abstract. We consider the bounded single-machine parallel-batch scheduling problem with release dates and rejection. A job is either rejected, in which case a certain penalty has to be paid, or accepted and then processed on the machine. The objective is to minimize the sum of the makespan of the accepted jobs and the total penalty of the rejected jobs. When the jobs have identical release dates, we present a polynomial-time algorithm. When the jobs have a constant number of release dates, we give a pseudo-polynomial-time algorithm. For the general problem, we provide a 2-approximation algorithm and a polynomial-time approximation scheme.

Keywords: scheduling; parallel-batch; rejection penalty; polynomial-time approximation scheme

1 Introduction

The bounded single-machine parallel-batch scheduling problem with release dates and rejection can be described as follows. There are n jobs $J_1, \ldots, J_n$ and a single machine that can process jobs in batches. Each job J_j has a processing time p_j, a release date r_j and a rejection penalty w_j. A job J_j is either rejected, in which case a certain penalty w_j has to be paid, or accepted and then processed on the machine. In the bounded model, the machine can process up to b jobs simultaneously as a batch. The processing time of a batch is defined as the longest processing time of the jobs contained in it. The objective is to minimize the sum of the makespan of the accepted jobs and the total penalty of the rejected jobs. Denote by W the total penalty of the rejected jobs. Using the general notation for scheduling problems, the problem is denoted by $1|\text{p-batch}, b < n, r_j|C_{\max} + W$.

In the last decade there has been significant interest in scheduling problems with batching considerations. The motivation for batching jobs is to gain in efficiency: it may be cheaper and faster to process jobs in a batch than to process them individually. The fundamental model of the parallel-batch scheduling problem was first introduced by Lee et al. [9] with the restriction that the number of the jobs in each batch is bounded by a number b. This bounded model is motivated by the burn-in operations in semiconductor manufacturing. Brucker et al. [1] provided an extensive discussion of the unbounded version of the parallel-batch scheduling problem. Recent developments on this topic can be found from the web site [2]. In addition, Liu and Yu [11], Deng and Zhang [4], Lee and Uzsoy [10], and Liu et al. [12] presented new complexity

*Corresponding author: T.C.E. Cheng. Email address: lgtcheng@polyu.edu.hk

results and approximation algorithms for the parallel-batch scheduling problem subject to release dates.

In the classical scheduling literature, all the jobs must be processed and no rejection is allowed. However, in real applications, this may not be true. Due to limited resources, the scheduler may have the option to reject some jobs. The machine scheduling problem with rejection was first considered by Bartal et al. [3]. They studied the off-line version, as well as the on-line version, of scheduling with rejection on identical parallel machines. The objective is to minimize the sum of the makespan of the accepted jobs and the total penalty of the rejected jobs. After that, the machine scheduling problem with rejection has received increasing research attention. Seiden [14] presented an improved on-line algorithm if preemption is allowed for all the jobs. Hoogeveen et al. [8] considered the off-line multi-processor scheduling problem with rejection where preemption is allowed. Engels et al. [5] studied single-machine scheduling with rejection to minimize the sum of the weighted completion times of the accepted jobs and the total penalty of the rejected jobs. Epstein et al. [6] considered on-line scheduling of unit-time jobs with rejection to minimize the total completion time.

For the problem $1|\text{p-batch}, b = \infty, r_j|C_{\max} + W$, Lu et al. [13] presented an NP-hardness proof, a 2-approximation algorithm, and a fully polynomial-time approximation scheme. In this paper we consider the scheduling problem $1|\text{p-batch}, b < n, r_j|C_{\max} + W$. Brucker et al. [1] showed that $1|\text{p-batch}, b < n, r_j|C_{\max}$ is strongly NP-hard. Thus $1|\text{p-batch}, b < n, r_j|C_{\max} + W$ is strongly NP-hard, too, and so has no fully polynomial-time approximation scheme if P≠NP. Liu and Yu [11] showed that $1|\text{p-batch}, b < n, r_j|C_{\max}$ is binary NP-hard even when there are two distinct release dates. This means that even when the jobs have a constant number of release dates, $1|\text{p-batch}, b < n, r_j|C_{\max} + W$ has no polynomial-time algorithm if P≠NP.

The main results in this paper are as follows. When the jobs have identical release dates, we present a polynomial-time algorithm. When the jobs have a constant number of release dates, we give a pseudo-polynomial-time algorithm. For the general problem, we provide a 2-approximation algorithm and a polynomial-time approximation scheme.

2 Exact algorithms

In this section we consider two special cases: (1) the case with identical release dates, and (2) the case with k distinct release dates, where k is a fixed positive integer. We provide a polynomial time algorithm for the first case and a pseudo-polynomial time algorithm for the second case.

2.1 The case with identical release dates

Assume that $r_j = r$ for $j = 1, \cdots, n$. For the bounded parallel-batch scheduling problem to minimize the makespan, which can be denoted by $1|\text{p-batch}, b < n|C_{\max}$, it can be solved optimally (see Lee and Uzsoy [10]) by the full batch longest processing time rule (FBLPT-rule).

FBLPT-rule:

Step 1: Index the jobs such that $p_1 \geq \cdots \geq p_n$.

Step 2. Assign all the jobs to batches in increasing order of their indexes such that each batch contains exactly b jobs apart from the last batch.

Step 3. Sequence the batches in an arbitrary order.

By the optimality of the FBLPT-rule for $1|\text{p-batch}, b < n|C_{\max}$, we have the following lemma.

Lemma 2.1.1. There exists an optimal schedule for $1|\text{p-batch}, b < n, r_j = r|C_{\max} + W$ in which the accepted jobs are assigned to the machine by the FBLPT-rule.

Assume that the jobs have been indexed such that $p_1 \geq \cdots \geq p_n$. Let $A_j(a)$ be the optimal value of the objective function satisfying the following conditions: (1) the jobs in consideration are $J_1, \cdots, J_j$, (2) J_j is accepted, and (3) the number of accepted jobs among $J_1, \cdots, J_j$ is exactly a. Similarly, let $R_j(a)$ be the optimal value of the objective function satisfying the following conditions: (1) the jobs in consideration are $J_1, \cdots, J_j$, (2) J_j is rejected, and (3) the number of accepted jobs among $J_1, \cdots, J_j$ is exactly a. We distinguish four cases in the following discussion.

Case 1. Both J_{j-1} and J_j are rejected.

Since J_j is rejected, the number of accepted jobs among $J_1, \cdots, J_{j-1}$ is still a. Thus, we have $R_j(a) = R_{j-1}(a) + w_j$ since J_{j-1} is rejected.

Case 2. J_{j-1} is accepted and J_j is rejected.

Similar to Case 1, we have $R_j(a) = A_{j-1}(a) + w_j$.

Case 3. J_{j-1} is rejected and J_j is accepted.

If $a = kb + 1$, then J_j has to start a new batch, and we have $A_j(a) = R_{j-1}(a-1) + p_j$. If $a \neq kb + 1$, then J_j can be assigned to the last batch, and we have $A_j(a) = R_{j-1}(a-1)$. In conclusion, we have

$$A_j(a) = \begin{cases} R_{j-1}(a-1) + p_j & \text{if } \ a = kb + 1; \\ R_{j-1}(a-1) & \text{otherwise.} \end{cases}$$

Case 4. Both J_{j-1} and J_j are accepted.

Similar to Case 3, we have

$$A_j(a) = \begin{cases} A_{j-1}(a-1) + p_j & \text{if } \ a = kb + 1; \\ A_{j-1}(a-1) & \text{otherwise.} \end{cases}$$

Combining the above four cases, we can devise the following dynamic programming algorithm DP1.

Dynamic programming algorithm DP1

The boundary conditions:

$$A_1(1) = r + p_1 \text{ and } A_1(a) = \infty \text{ for any } a \neq 1.$$

$$R_1(0) = w_1 \text{ and } R_1(a) = \infty \text{ for any } a \neq 0.$$

The recursive function:

$$A_j(a) = \begin{cases} \min\{A_{j-1}(a-1)+p_j, R_{j-1}(a-1)+p_j\} & \text{if } a = kb+1; \\ \min\{A_{j-1}(a-1), R_{j-1}(a-1)\} & \text{otherwise.} \end{cases}$$

$$R_j(a) = \min\{A_{j-1}(a), R_{j-1}(a)\} + w_j.$$

The optimal value is given by $\min\{\min\{A_n(a), R_n(a)\} : 0 \le a \le n\}$.

Theorem 2.1.2. DP1 solves $1|\text{p-batch}, b < n, r_j = r|C_{\max} + W$ in $O(n^2)$ time.

Proof. The correctness of the algorithm is guaranteed by the above discussion. The recursive function has at most $O(n^2)$ states. Each iteration takes a constant time to execute. Hence the time complexity is bounded by $O(n^2)$. □

2.2 The case with k distinct release dates

For the problem $1|\text{p-batch}, b < n, r_j|C_{\max}$ with a constant number of release dates, Liu and Yu [11] presented a pseudo-polynomial-time algorithm. By extending their algorithm, we present the following pseudo-polynomial-time algorithm for $1|\text{p-batch}, b < n, r_j|C_{\max} + W$ with a constant number of release dates.

Assume that the jobs have been indexed such that $p_1 \ge \cdots \ge p_n$. Let $R_1, R_2, \cdots, R_k$ with $R_1 < R_2 < \cdots < R_k$ be the k distinct release dates. We divide $[R_1, +\infty)$ into k time intervals $[R_1, R_2), [R_2, R_3), \cdots, [R_k, R_{k+1})$, where $R_{k+1} = +\infty$. Let $f_j(s_1, \cdots, s_k; a_1, \cdots, a_k; l_1, \cdots, l_k; w)$ be the optimal value of the objective function satisfying the following conditions: (1) The jobs in consideration are $J_1, \cdots, J_j$. (2) The first batch starting in $[R_i, R_{i+1})$ starts at time s_i. If no batch starts in $[R_i, R_{i+1})$, then we set $s_i = +\infty$. (3) The last batch starting in $[R_i, R_{i+1})$ contains a_i jobs. If no batch starts in $[R_i, R_{i+1})$, then we set $a_i = b$. (4) The total length of the batches starting in $[R_i, R_{i+1})$ is l_i. If no batch starts in $[R_i, R_{i+1})$, then we set $l_i = 0$. (5) The total penalty of rejected jobs is exactly w. We distinguish two cases in the following discussion.

Case 1: J_j is rejected.

Since J_j is rejected, we have $f_j(s_1, \cdots, s_k; a_1, \cdots, a_k; l_1, \cdots, l_k; w) = f_{j-1}(s_1, \cdots, s_k; a_1, \cdots, a_k; l_1, \cdots, l_k; w - w_j) + w_j$.

Case 2: J_j is accepted.

Given a state vector $(s_1, \cdots, s_k; a_1, \cdots, a_k; l_1, \cdots, l_k; w)$, we assume that $s_l = \max\{s_i : s_i \ne +\infty\}$. Let $h_i(j)$ be the optimal objective value under the constraint that the batch containing J_j starts in $[R_i, R_{i+1})$, where $R_i \ge r_j$ and $i \le l$. If $a_i = 1$ and $i < l$, then J_j has to start a new batch in $[R_i, R_{i+1})$ and this does not increase the makespan of the accepted jobs since $i < l$. Thus, we have $h_i(j) = f_{j-1}(s_1, \cdots, s_k; a_1, \cdots, a_{i-1}, b, a_{i+1}, \cdots, a_k; l_1, \cdots, l_{i-1}, l_i - p_j, l_{i+1}, \cdots, l_k; w)$. If $a_i = 1$ and $i = l$, then J_j has to start a new batch in $[R_i, R_{i+1})$ and this will increase the makespan of the accepted jobs by a length p_j since $i = l$. Thus, we have $h_i(j) = f_{j-1}(s_1, \cdots, s_k; a_1, \cdots, a_{i-1}, b, a_{i+1}, \cdots, a_k; l_1, \cdots, l_{i-1}, l_i - p_j, l_{i+1}, \cdots, l_k; w) + p_j$. If $a_i > 1$, then J_j can be assigned in the last batch in $[R_i, R_{i+1})$. Thus, we have $h_i(j) = f_{j-1}(s_1, \cdots, s_k; a_1, \cdots, a_{i-1}, a_i - 1, a_{i+1}, \cdots, a_k; l_1, \cdots, l_k; w)$. Furthermore, we also have

$$f_j(s_1,\cdots,s_k;a_1,\cdots,a_k;l_1,\cdots,l_k;w)=\min\{h_i(j): R_i\geq r_j \text{ and } 1\leq i\leq l\},$$

where

$$h_i(j)=\begin{cases} f_{j-1}(s_1,\cdots,s_k;a_1,\cdots,a_{i-1},b,a_{i+1},\cdots,a_k;l_1,\cdots,l_{i-1},l_i-p_j,l_{i+1},\cdots,l_k;w) \\ \qquad\qquad \text{if } \ a_i=1 \text{ and } i<l; \\ f_{j-1}(s_1,\cdots,s_k;a_1,\cdots,a_{i-1},b,a_{i+1},\cdots,a_k;l_1,\cdots,l_{i-1},l_i-p_j,l_{i+1},\cdots,l_k;w)+p_j \\ \qquad\qquad \text{if } \ a_i=1 \text{ and } i=l; \\ f_{j-1}(s_1,\cdots,s_k;a_1,\cdots,a_{i-1},a_i-1,a_{i+1},\cdots,a_k;l_1,\cdots,l_k;w) \\ \qquad\qquad \text{if } \ a_i>1. \end{cases}$$

Combining the above two cases, we design the following dynamic programming algorithm DP2.

Dynamic programming algorithm DP2

The boundary conditions:

Given a state vector $(s_1,\cdots,s_k;a_1,\cdots,a_k;l_1,\cdots,l_k;w)$, we assume that $s_l=\max\{s_i: s_i\neq+\infty\}$. We define $f_0(s_1,\cdots,s_k;b,\cdots,b;0,\cdots,0;0)=s_l$, and for any otherwise cases, $f_0(s_1,\cdots,s_k;a_1,\cdots,a_k;l_1,\cdots,l_k;w)=+\infty$.

The recursive function:

$f_j(s_1,\cdots,s_k;a_1,\cdots,a_k;l_1,\cdots,l_k;w)=\min\{f_{j-1}(s_1,\cdots,s_k;a_1,\cdots,a_k;l_1,\cdots,l_k;w-w_j)+w_j,\ \min\{h_i(j): R_i\geq r_j \text{ and } 1\leq i\leq l\}\}$, where

$$h_i(j)=\begin{cases} f_{j-1}(s_1,\cdots,s_k;a_1,\cdots,a_{i-1},b,a_{i+1},\cdots,a_k;l_1,\cdots,l_{i-1},l_i-p_j,l_{i+1},\cdots,l_k;w) \\ \qquad\qquad \text{if } \ a_i=1 \text{ and } i<l; \\ f_{j-1}(s_1,\cdots,s_k;a_1,\cdots,a_{i-1},b,a_{i+1},\cdots,a_k;l_1,\cdots,l_{i-1},l_i-p_j,l_{i+1},\cdots,l_k;w)+p_j \\ \qquad\qquad \text{if } \ a_i=1 \text{ and } i=l; \\ f_{j-1}(s_1,\cdots,s_k;a_1,\cdots,a_{i-1},a_i-1,a_{i+1},\cdots,a_k;l_1,\cdots,l_k;w) \\ \qquad\qquad \text{if } \ a_i>1. \end{cases}$$

The optimal value is given by $\min\{f_n(s_1,\cdots,s_k;a_1,\cdots,a_k;l_1,\cdots,l_k;w)\}$.

Theorem 2.2.1. DP2 solves $1|\text{p-batch},b<n,r_j\in\{R_i: 1\leq i\leq k\}|C_{\max}+W$ in $O(nkb^kp_1^{k-1}(\sum w_j)(\sum p_j)^k)$ time.

Proof. The correctness of the algorithm is guaranteed by the above discussion. Clearly, we have $1\leq a_i\leq b$, $0\leq l_i\leq\sum p_j$ and $0\leq w\leq\sum w_j$. Furthermore, if some batch starts in $[R_i,R_{i+1})$ for $i=1,\cdots,k$, then we have $s_1=R_1$ and $R_i\leq s_i<\min\{R_i+p_1,R_{i+1}\}$. Thus, the recursive function has at most $O(nb^kp_1^{k-1}(\sum w_j)(\sum p_j)^k)$ states. Each iteration takes an $O(k)$ time to execute. Hence the total running time is bounded by $O(nkb^kp_1^{k-1}(\sum w_j)(\sum p_j)^k)$. □

3 Approximation algorithms

In this section we present a 2-approximation algorithm and a polynomial-time approximation scheme for $1|\text{p-batch}, b < n, r_j|C_{\max} + W$.

3.1 A 2-approximation algorithm

Assume that S is a set of jobs. We use $p(S) = \max_{J_j \in S} p_j$ and $w(S) = \sum_{J_j \in S} w_j$ to denote the processing time and the total rejection penalty of S, respectively. We propose a 2-approximation algorithm for the considered problem.

Approximation algorithm A

Step 1. For each $t \in \{r_j : j = 1, \ldots, n\}$, we divide the jobs into two sets of jobs such that $S_1(t) = \{J_j : r_j \leq t\}$ and $S_2(t) = \{J_j : r_j > t\}$.

Step 2. Setting $r_j = 0$ for each $J_j \in S_1(t)$, we obtain a new instance $I(t)$. Apply algorithm DP1 to the instance $I(t)$. Let $B_1, \cdots, B_k$ be the batches of the accepted jobs obtained from DP1. Process $B_1, \cdots, B_k$ from time t in an arbitrary order on the machine and reject all the other jobs. The resulting schedule for the original instance is denoted by $\pi(t)$.

Step 3. Let $Z(t)$ be the value of the objective function for each $\pi(t)$. Among all the schedules obtained above, select the one with the minimum $Z(t)$ value. □

Let π be the schedule obtained by the approximation algorithm A. Let Z and Z^* be the objective values of the schedule π and an optimal schedule π^*, respectively.

Theorem 3.1.1. $Z \leq 2Z^*$ and the bound is tight.

Proof. Let A^* and R^* be the sets of accepted and rejected jobs in π^*, respectively. Let $r^* = \max\{r_j : J_j \in A^*\}$. By the definition of r^*, we have $S_2(r^*) = \{J_j : r_j > r^*\} \subseteq R^*$. Then we have $Z^* \geq r^* + w(S_2(r^*))$. Apply algorithm DP1 to the instance $I(r^*)$. Let $B_1, \cdots, B_k$ be the batches of accepted jobs and let R be the set of rejected jobs obtained from DP1. Clearly, we also have $Z^* \geq Z(r^*) - r^* = p(B_1) + \cdots + p(B_k) + w(R)$. Thus, we have

$$Z \leq Z(r^*) = r^* + p(B_1) + \cdots + p(B_k) + w(R) + w(S_2(r^*)) \leq 2Z^*.$$

To show that the bound is tight, we consider the following instance with three jobs: Let $b = 1$. $(r_1, p_1, w_1) = (0, 1, 2)$, $(r_2, p_2, w_2) = (0, 1, 0)$ and $(r_3, p_3, w_3) = (1, 0, 2)$. It is easy to verify that $Z(0) = p_1 + w_2 + w_3 = 3$ and $Z = Z(1) = 1 + p_1 + p_3 + w_2 = 2$. However, the optimal schedule is to accept J_1, J_3 and reject J_2, with J_1 starting processing at time 0 followed by J_3. That is, $Z^* = 1$. Thus, we have $Z = 2 = 2Z^*$. □

3.2 A polynomial-time approximation scheme

Let Z and Z^* be the objective values of the approximation algorithm A and an optimal schedule π^*, respectively. By Theorem 3.1.1, we have $Z^* \leq Z \leq 2Z^*$. For any job J_j with $w_j > Z$, J_j

must be accepted in π^*. Otherwise, we have $Z^* \geq w_j > Z \geq Z^*$, a contradiction. Similarly, for any job J_j with $r_j > Z$ or $p_j > Z$, J_j must be rejected in π^*. We modify r_j, p_j and w_j such that $r_j = \min\{r_j, Z\}$, $p_j = \min\{p_j, Z\}$ and $w_j = \min\{w_j, Z\}$. Clearly, this does not change the optimal objective value. Thus, we can assume that $\max\{r_j, p_j, w_j\} \leq Z$ for each $j = 1, \cdots, n$. Now, we propose a polynomial-time approximation scheme A_ϵ for this problem.

Polynomial-time approximation scheme A_ϵ

Step 1. For any $\epsilon > 0$, set $\delta = \epsilon Z$ and $M = \frac{\epsilon Z}{2n}$. Given an instance I, we define a new instance I' by rounding r_j, p_j and w_j in I such that $r'_j = \lfloor \frac{r_j}{\delta} \rfloor \delta$, $p'_j = \lfloor \frac{p_j}{M} \rfloor M$ and $w'_j = \lfloor \frac{w_j}{M} \rfloor M$, for $j = 1, \cdots, n$.

Step 2. Apply algorithm DP2 to the instance I' to obtain an optimal solution $\pi^*(I')$ for the instance I'.

Step 3. Increase the starting time of each job in $\pi^*(I')$ by δ and replace p'_j and w'_j by the original p_j and w_j in $\pi^*(I')$, respectively, for each $j = 1, \cdots, n$, to obtain a feasible solution π for the instance I.

Let Z_ϵ be the objective value of the schedule π obtained from A_ϵ. We have the following theorem.

Theorem 3.2.1 Algorithm A_ϵ is a polynomial-time approximation scheme for the problem $1|\text{p-batch}, b < n, r_j|C_{\max} + W$. Specifically, A_ϵ is a fully polynomial-time approximation scheme when there are k distinct release dates in the original instance, where k is a fixed positive integer.

Proof. Let $Z^*(I')$ be the optimal objective value of the schedule $\pi^*(I')$. Clearly, we have $Z^*(I') \leq Z^*$. Increase the starting time of each job in $\pi^*(I')$ by $\delta \leq 2\epsilon Z^*$, which increases the objective value by at most $2\epsilon Z^*$. Replace p'_j and w'_j by p_j and w_j, respectively, for each $j = 1, \cdots, n$. It is easy to see that we obtain a feasible schedule for the instance I. Thus, we have

$$Z_\epsilon \leq Z^*(I') + 2\epsilon Z^* + \sum_{j=1}^{n}(p_j - p'_j) + \sum_{j=1}^{n}(w_j - w'_j) \leq Z^* + 2\epsilon Z^* + nM + nM \leq (1 + 4\epsilon)Z^*.$$

Since $r_j \leq Z$, there are at most $\frac{1}{\epsilon}+1$ distinct release dates in I'. Since $p_j \leq Z$ for $j = 1, \cdots, n$, we have $\sum_{j=1}^{n} \lfloor \frac{p_j}{M} \rfloor \leq \frac{2n}{\epsilon} \sum_{j=1}^{n} \frac{p_j}{Z} \leq \frac{2n^2}{\epsilon}$. Similarly, we have $\sum_{j=1}^{n} \lfloor \frac{w_j}{M} \rfloor \leq \frac{2n}{\epsilon} \sum_{j=1}^{n} \frac{w_j}{Z} \leq \frac{2n^2}{\epsilon}$. Thus, the time complexity of DP2 (also A_ϵ) is $O(nkb^k p_1^{k-1}(\sum w_j)(\sum p_j)^k) = O(n(\frac{1}{\epsilon} + 1)b^{\frac{1}{\epsilon}+1}(\frac{2n^2}{\epsilon})^{\frac{2}{\epsilon}+2})$, confirming that algorithm A_ϵ is a polynomial-time approximation scheme. Specifically, if there are k distinct release dates in the orginal instance I, then the rounding stance I' has at most k distinct release dates. Thus, the time complexity of DP2 (also A_ϵ) is $O(nkb^k p_1^{k-1}(\sum w_j)(\sum p_j)^k) = O(nkb^k(\frac{2n^2}{\epsilon})^{2k})$. That is, A_ϵ is a fully polynomial-time approximation scheme when there are k distinct release dates in the original instance. □

Acknowledgments

This research was supported in part by the Research Grants Council of Hong Kong under grant number N-PolyU502/07. Yuan and Lu were also supported in part by grants NSFC (10671183) and NFSC-RGC (70731160633).

References

[1] P. Brucker, A. Gladky, H. Hoogeveen, M.Y. Kovalyov, C.N. Potts, S.L. van de Velde, Scheduling a batching machine. Journal of scheduling 1, 31-54, 1998.

[2] P. Brucker, S. Knust, Complexity results for scheduling problem, http://www.mathematik.uniosnabrueck.de/reseach/OR/class/2007.

[3] Y. Bartal, S. Leonardi, A. M. Spaccamela, J. Sgall, L. Stougie, Multiprocessor scheduling with rejection. SIAM Journal on Discrete Mathematics 13, 64-78, 2000.

[4] X. Deng, Y.Z. Zhang, Minimizing mean response time for batch processing systems. Lecture Notes on Computer Science 1627, 231-240, 1999.

[5] D.W. Engels, D.R. Karger, S.G. Kolliopoulos, S. Sengupta, R.N. Uma, J. Wein, Techniques for scheduling with rejection. Journal of Algorithms 49, 175-191, 2003.

[6] L. Epstein, J. Noga, G.J. Woeginger, On-line scheduling of unit time jobs with rejection: minimizing the total completion time. Operations Research Letters 30, 415-420, 2002.

[7] R.L. Graham, E.L. Lawer, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics 5, 1-15, 1979.

[8] H. Hoogeveen, M. Skutella, G.J. Woeginger, Preemptive scheduling with rejection. Mathematical Programming 94, 361-374, 2003.

[9] C.-Y. Lee, R. Uzsoy, L.A. Martin-Vega, Efficient algorithms for scheduling semiconductor burn-in operations. Opereations Research 40, 764-775, 1992.

[10] C.-Y. Lee, R. Uzsoy, Minimizing makespan on a single batch processing machine with dynamic job arrivals. International Journal of Production Research 37, 219-236, 1999.

[11] Z.H. Liu, W.C. Yu, Scheduling one batch processor subject to job release dates. Discrete Applied Mathematics 105, 129-136, 2000.

[12] Z.H. Liu, J.J. Yuan, T.C.E. Cheng, On scheduling an unbounded batch machine. Operations Research Letters 31, 42-48, 2003.

[13] L.F. Lu, L.Q. Zhang, J.J. Yuan, 2008. The unbounded parallel batch machine scheduling with release dates and rejection to minimize makespan. Theoretical Computer Science 396, 283-289.

[14] S. Seiden, Preemptive multiprocessor scheduling with rejection. Theoretical Computer Science 262, 437-458, 2001.